Docket No. RSW920030248US1 (7161-133U) **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of	:	Customer Number: 46320
David GILGEN, et al.	:	Confirmation Number: 9498
Application No.: 10/723,979	:	Group Art Unit: 2191
Filed: November 26, 2003	:	Examiner: A. Deng

For: FAST DETECTION OF THE ORIGINS OF MEMORY LEAKS WHEN USING POOLED RESOURCES

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Appeal Brief is submitted in support of the Notice of Appeal filed October 24, 2007, wherein Appellants appeal from the Examiner's rejection of claims 1-20.

I. REAL PARTY IN INTEREST

This application is assigned to IBM Corporation by assignment recorded on November 26, 2003, at Reel 014755, Frame 0965.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals and interferences.

III. STATUS OF CLAIMS

Claims 1-20 are pending and two-times rejected in this Application. It is from the multiple rejections of claims 1-20 that this Appeal is taken.

IV. STATUS OF AMENDMENTS

The claims have not been amended subsequent to the imposition of the Second Office Action dated July 24, 2007 (hereinafter the Second Office Action).

V. SUMMARY OF CLAIMED SUBJECT MATTER

Referring to Fig. 2 and to independent claims 1 and 15, a memory leak detection and reporting method is disclosed. In block 240, allocated ones of resources in a resource pool are time stamped (lines 8-9 of paragraph [0023] of Appellants' disclosure). In block 260, calling code segments receiving the allocated resources are identified (lines 20-12 of paragraph [0023]). In blocks 280 and 290, memory leaks are detected by inspecting individual timestamps for the allocated resources to determine whether the allocated resources have become overly idle (lines 1-4 and 6-8 of paragraph [0024]). In block 310, for each allocated resource determined to have become overly idle, an identity of a corresponding one of the calling code segments is reported (lines 1-4 of paragraph [0025]).

Referring to Fig. 1 and to independent claim 7, a memory leak detection and reporting system is disclosed. The system includes a resource pool 110, a pool manager 130, and a data store 140. The resource pool 110 includes a plurality of allocable resources 170 (lines 4-6 of paragraph [0016]). The pool manager 130 is programmed to manage allocation of the allocable resources 170 to calling code segments 150 (lines 3-4 of paragraph [0016]). The data store 140 include allocated resources 170 and corresponding identities for calling code segments 150

receiving the allocated resources 170 (lines 1-8 of paragraph [0018]). The pool manager 130 also detects memory leaks by inspecting individual timestamps 180 for the allocated resources 170 to determine whether the allocated resources 170 have become overly idle and for each allocated resource 170 determined to have become overly idle, reports an identity of a corresponding one of the calling code segments 150 to the data store 140 (lines 4-12 of paragraph [0021]).

Referring to Fig. 2 and to independent claims 10 and 20, a memory leak detection and reporting method is disclosed. In block 230, a resource from a resource pool is allocated (lines 7-8 of paragraph [0023]). In block 240, the allocated resource is time stamped (lines 8-9 of paragraph [0023]). In block 260, an identity for a calling code segment acquiring the allocated resource is recorded (lines 10-12 of paragraph [0023]). When the allocated resource is accessed, the time stamp is updated (lines 1-3 of paragraph [0024]). In blocks 280 and 290, the time stamp is inspected to determine if the allocated resource has become overly idle (lines 1-4 and 6-8 of paragraph [0024]). In block 310, if it is determined that the allocated resource has become overly idle, a suspected memory leak in association with the allocated resource is reported and the recorded identity for the calling code segment which had acquired the allocated resource is reported (lines 1-4 of paragraph [0025]).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 7-9 are rejected under 35 U.S.C. § 101;

1. Claims 1-3, 5-17, and 19-20 were rejected under 35 U.S.C. § 103 for obviousness based upon Dahlstedt et al., U.S. Patent Publication No. 2004/0133894 (hereinafter Dahlstedt) in view of Tarditi, U.S. Patent No. 6,625,808; and

2. Claims 4 and 18 were rejected under 35 U.S.C. § 103 for obviousness based upon Dahlstedt in view of Tarditi and further in view of Fu, U.S. Patent Publication No. 2004/0172579.

VII. ARGUMENT

THE REJECTION OF CLAIMS 7-9 UNDER 35 U.S.C. § 101

For convenience of the Honorable Board in addressing the rejections, and claims 8 and 9 stand or fall together with independent claim 7.

On pages 7 and 8 of the First Amendment dated May 31, 2007 (hereinafter the First Amendment), Appellants presented the following arguments. On page 2 of the Office Action, the Examiner asserted the following with regard to claims 7-9:

> Claims 7-9 set forth a memory leak detection and reporting system that is computer program claimed as computer listings per se, i.e., the descriptions or expressions of the programs, are not physical "things." They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer which permit the computer program's functionality to be realized.

The Examiner's analysis appears to be predicated on the belief that the claimed memory leak detection and reporting system is "computer listings per se." In this regard, Appellants respectfully submit that the Examiner's analysis is flawed. Based upon the Examiner's analysis, it appears that the Examiner misunderstands exactly what constitutes software (i.e., computer listings) *per se*. In particular, Appellants respectfully submit that the Examiner is confused as to exact meaning of the phrase "per se." The definition of "per se" is the following:[1]

> By itself; in itself; taken alone; by means of itself; through itself; inherently; in isolation; unconnected with other matters; simply as such; in its own natures without reference to its relation.

Thus, software *per se* is an abstract idea embodied by the software alone without anything else. For this reason, software *per se* is deemed to be non-statutory subject matter.

[1] Black's Law Dictionary 1142 (6th ed. 1990).

Software alone is incapable of doing anything because it is disconnected from hardware. Software alone is also covered under M.P.E.P. § 2106.01 with regard to "functional descriptive material." Claims 7-9, however, are not directed to "functional descriptive material." Instead, claims 7-9 are directed to a system.

Claim 7 recites "a pool manager programmed ..." (emphasis added). Software alone is incapable of being "programmed." Instead, hardware (i.e., a device) is programmed, and software alone is the program. Moreover, Appellants are entirely unclear as to why the claimed "data store" is also not considered hardware. Software alone is incapable of storing anything since it is functional descriptive material or an abstract idea.

On page 11 of the Second Office Action, the Examiner responded to only one of these arguments in asserting the following:

> The data store as recited in Specification, "The data store of allocated resources 140 can include a listing of all resources in the resource pool 110 which have been allocated previously to calling code segments ... (specification, p. 10, [0018])", that does connect to hardware (i.e., a device, a physical memory). Data store can be software (i.e., database, data structure). There is no description to specify the "data store" is a software or hardware in the Specification or Drawing. It appears to the examiner, the system in Claims 5-8 is a software system.

The Examiner's response doesn't address the substance of Appellants' prior arguments. For example, the Examiner has not explained how a "data store" can store data without being connected to hardware. Appellants are unaware that abstract ideas (i.e., software *per se*) are capable of storing data. Similarly, abstract ideas are also incapable of being programmed.

Despite the Examiner's continued failure to set forth an explanation as to why these claimed elements are directed to an abstract idea, reference is made to the recent decision of the

Federal Circuit of In re Comiskey.[2] Although the Court held that several claims were directed to non-statutory subject matter, the Court determined that other of the claims were directed to statutory subject matter. In determining the latter, the Court stated the following:

> These claims, under the broadest reasonable interpretation, could require the use of a computer as part of Comiskey's arbitration system. (emphasis added)

Thus, the Court determined that the claims are not required to necessarily include or require a computer. Instead, the Court concluded that the claims meet the requirements of 35 U.S.C. § 101 if, under a broadest reasonable interpretation, the claims could require the use of statutory subject matter (e.g., a computer, a device, a product, etc.).

it is readily apparent that under a *broadest reasonable interpretation*, the claimed data store could require the use of a computer storage device (e.g., memory, hard drive, etc.) and that since the pool manager is programmed, the pool manager could be a computer device. Thus, based upon the analysis in In re Comiskey, the claimed invention, as recited in claim 7, is directed to statutory subject matter.

THE REJECTION OF CLAIMS 1-3, 5-17, AND 19-20 UNDER 35 U.S.C. § 103 FOR OBVIOUSNESS BASED UPON DAHLSTEDT IN VIEW OF TARDITI

For convenience of the Honorable Board in addressing the rejections, claims 2, 6-10, 12, 14-16, and 19-20 stand or fall together with independent claim 1; claims 11 and 17 stand or fall together with dependent claim 3; and claim 13 stands or falls with dependent claim 5

[2] Appeal No. 2006-1286.

Claim 1

On page 9 and 10 of the First Amendment, Appellants presented the following arguments. On page 3 of the First Office Action, the Examiner asserted the following with regard to the teachings of Dahlstedt:

> reporting an identity of a corresponding one of said calling code segments (Dahlstedt, [0008], "the invention comprises a system for determining potential memory leaks in a run-time environment, ... an object temperature analyzer that determines the status of warm objects and cold objects in said memory, and the links between said warm and cold objects, and, a report mechanism that reports information about said links, for use in determining potential memory leaks", also see, FIG. 3, [0021], Dahlstedt teaches cold objects as determined to have become overly idle, "The time stamp for each object is checked against the current system time, and those objects that have a time stamp older than a particular period of time are marked as cold objects").

The Examiner further asserted the following on page 4 of the First Office Action with regard to the teachings of Tarditi and what Dahlstedt fails to teach:

> Dahlstedt does not explicitly teach identifying calling code segments receiving said allocated resources. However, Tarditi teaches identifying calling code segments receiving said allocated resources (Tarditi, FIG. 4, call stack 402 or 406, col. 11, lines 10-49, "a live object is an object which has an identifiable pointer in the root set, e.g., call stack 402 or 406... for each transition from a GC frame to a non-GC frame in the call stack 402, creation function 304 allocates space on the stack frame for a transition record, e.g., transition records 420 and 424. The transition store select pointer and state information, as well as a pointer to the immediate past transition record, e.g., transition record 424.").

The Examiner's analysis present a logical inconsistency. The Examiner's admits that Dahlstedt does not teach the claimed "identifying calling code segments receiving said allocated resources." However, the Examiner asserts that Dahlstedt teaches "reporting an identity of a corresponding one of said calling code segments." The logical inconsistency of the Examiner's analysis can be best summarized by the question: how can Dahlstedt teach reporting an identity of a calling code segment when Dahlstedt does not teach identifying the calling code segment? The calling code segment must first be identified before its identity can be reported. Therefore, the Examiner's assertion that Dahlstedt teaches "reporting an identity of a corresponding one of

said calling code segments" cannot be true since Dahlstedt has not identified the calling code segments.

In response, on page 3 of the Second Office Action, the Examiner is no longer asserting that Dahlstedt teaches "reporting an identity of a corresponding one of said calling code segments." Instead, the Examiner is now relying upon Tarditi to teach this limitation. Specifically, the Examiner is now asserting the following on page 4 of the Second Office Action:

> Dahlstedt does not explicitly teach identifying calling code segments receiving said allocated resources; and identity of a corresponding one of said calling code segments. However, Tarditi teaches identifying calling code segments receiving said allocated resources, reporting an identity of a corresponding one of said calling code segments. (Tarditi, FIG, 4, call stack 402 or 406, col. 11, lines 10-49, "a live object is an object which has an identifiable pointer in the root set, e.g., call stack 402 or 406... for each transition from a GC frame to a non-GC frame in the call stack 402, creation function 304 allocates space on the stack frame for a transition record, e.g., transition records 420 and 424. The transition store select pointer and state information, as well as a pointer to the immediate past transition record, e.g., transition record 424."), the transition record read in the limitation of reporting identity of a corresponding one of said calling code segments. (emphasis in original)

The Examiner's analysis, however, fails to consider the claimed invention, as a whole. Specifically, claim 1 recites:

> *for each allocated resource determined to have become overly idle*, reporting an identity of a corresponding one of said calling code segments (emphasis added).

As readily evident upon reviewing this passage, the second clause (i.e., the underlined clause) is conditioned upon the first clause (i.e., the *italicized* clause). Thus, there is an inextricable relationship between the first clause and the second clause. The Examiner's analysis, however, relies upon Dahlstedt to teach the limitations of the first clause and Tarditi to teach the limitations of the second clause. In so doing, the Examiner has failed to consider the claimed invention, as a whole. Instead, the Examiner has taken teachings from two separate references and linked them

together, in the manner claimed, without any common sense rationale for making this linkage. Appellants position is that the only rationale of record for making this linkage comes from Appellants' own disclosure, and for the Examiner to rely on this rationale constitutes impermissible hindsight reconstruction of the claimed invention.

On pages 10 and 11 of the First Amendment, Appellants presented the following arguments regarding the Examiner's asserted rationale to modify Dahlstedt in view of Tarditi. Specifically, with regard to the Examiner's asserted rationale to modify Dahlstedt in view of Tarditi, the Examiner asserted the following in the first full paragraph on page 4 of the First Office Action:

> It would have been obvious to one having ordinary skill in the computer art at the time of the invention was made to modify the method disclosed by Dahlstedt to include identifying calling code segments receiving said allocated resources using the teaching of Tarditi. The modification would be obvious because one of ordinary skill in the art would be motivated to facilitate automated memory management among heterogeneous components of a computer program (Tarditi, col. 4, lines 40-42).

The Examiner's asserted benefit is little more than a generalization regarding a possible benefit resulting from the entirety of the teachings of Tarditi. The Examiner's analysis, however, has failed to establish that this particular benefit would result solely from the Examiner's proposed modification to Dahlstedt. In this regard, Appellants respectfully submit that the Examiner has failed to establish a nexus between the specific teachings relied upon within Tarditi for the proposed modifications to Dahlstedt and the asserted benefit to "facilitate automated memory management among heterogeneous components of a computer program."

A nexus is required between the proposed modification and the asserted benefit of the modification. The need for a nexus between the proposed modification and the asserted benefit of the modification is to establish that one having ordinary skill in the art would have been

realistically impelled to modify the prior art in the manner suggested by the Examiner. Otherwise, the Examiner could assert that any possible modification taught by Tarditi could be based upon any possible benefit taught by Tarditi.

In the Second Office Action, the Examiner did not directly respond to these arguments. Instead, the Examiner simply rephrased the asserted rationale to modify Dahlstedt in view of Tarditi to include the additional limitation the Examiner is now relying upon Tarditi to teach. Specifically, the Examiner asserted the following on page 4 of the Second Office Action:

> It would have been obvious to one having ordinary skill in the computer art at the time of the invention was made to modify the method disclosed by Dahlstedt to include identifying calling code segments receiving said allocated resources; reporting an identity of a corresponding one of said calling code segments using the teaching of Tarditi. The modification would be obvious because one of ordinary skill in the art would be motivated to facilitate automated memory management among heterogeneous components of a computer program (Tarditi, col. 4, lines 40-42).

As readily apparent from comparing the Examiner's asserted rationales for making the proposed modification in the First and Second Office Actions, the Examiner has not addressed Appellants' previously presented arguments. Instead, the Examiner continues to rely upon a generalization (i.e., "to facilitate automated memory management among heterogeneous components of a computer program") without any explanation as to why the Examiner's proposed rationale would lead to the claimed limitations or without an analysis as to why one having ordinary skill in the art would only have to rely upon "common sense" to make this proposed combination. Therefore, Appellants maintain that the Examiner has failed to establish that one having ordinary skill in the art would have considered the claimed invention to be an obvious combination of the teachings of Dahlstedt and Tarditi.

Claim 3

On page 11 of the First Amendment, Appellants presented the following arguments regarding claim 3. In the First Office Action, the Examiner asserted that column 11, lines 2-7 of Tarditi teaches the claimed "performing said detecting and reporting steps in a separate thread of execution." Upon review, Appellants note that this passage refers to two separate threads. Appellants, therefore, are unclear as to where, specifically, Tarditi teaches that the detecting and reporting steps are performed in a separate thread of execution.

The Examiner's responded to these arguments on page 12 of the Second Office Action, in which the Examiner asserted the following:

> Tarditi teaches detecting and reporting steps in a separate thread of execution (Tarditi, col. 10, lines 11-51, "when creation function 304 is invoked, it creates a transition record on the stack frame of the program thread in which the foreign function call is found, and loads a pointer to the last transition record from the per-thread state. ... the stack frame for each thread with transition records providing the necessary pointer and state information to traverse any number of foreign function call embedded within the program", also, col. 11, lines 2-11, "the data structures include call stacks 402 and 406 associated with two separate threads of program execution , emphasis added), Tarditi teaches a transition record read in the limitation detecting and reporting steps in separate thread of execution. (emphasis in original)

Like much of the Examiner's analysis, the Examiner is entirely unclear as to what specific elements in the prior art are being relied upon to teach the specific elements of the claims. Notwithstanding this ambiguity, reference is made to the last underlined clause in the above-reproduced passage. Specifically, the Examiner emphasizes "two separate threads of program execution." Notably, the Examiner appears to have misunderstood the plain language of claim 3, which recites that the detecting and reporting steps are performed in a separate thread of execution (i.e., a single thread, which is separate) and not separate threads of execution (i.e., two separate threads). Since Tarditi is being relied upon to teach separate threads, then Tarditi fails to teach the limitations recited in claim 3.

Claim 5

On page 11 of the First Amendment, Appellants presented the following arguments regarding claim 5. The Examiner cited Fig. 4 and paragraph [0022] of Dahlstedt to teach the claimed "performing said detecting and reporting steps responsive to allocating one of said resources in said resource pool." As already noted above, however, Dahlstedt fails to teach the claimed reporting step. Thus, Dahlstedt cannot teach all the limitations of claim 5.

The Examiner's responded to these arguments on pages 12 and 13 of the Second Office Action, in which the Examiner asserted the following:

> Tarditi teaches the step of performing said detecting and reporting steps responsive to allocating one of said resources in said resource pool (Tarditi, col. lines 23-45, "when execution of the program proceeds to a creation function, the creation function creates a transition record on the stack frame of the particular thread in which the function call is identified. The transition record is populated with select pointer and state information").

Upon reviewing this passage, Appellants are unclear as to the relationship between column 9, lines 23-45 of Tarditi and the claimed "performing said detecting and reporting steps." Again, how the Examiner is interpreting the language of the claims and what specific teachings in the prior art the Examiner is relying upon to disclose these limitations is unclear.

Appellants, therefore, respectfully submit that the imposed rejection of claims 1-3, 5-17, and 19-20 under 35 U.S.C. § 103 for obviousness based upon Dahlstedt in view of Tarditi is not viable.

THE REJECTION OF CLAIMS 4 AND 18 UNDER 35 U.S.C. § 103 FOR OBVIOUSNESS BASED UPON DAHLSTEDT IN VIEW OF TARDITI AND FU

For convenience of the Honorable Board in addressing the rejections, and claim 18 stands or falls together with dependent claim 4.

On page 13 of the First Amendment, Appellants noted that upon reviewing the teachings in Fu cited by the Examiner, Appellants are entirely unclear where Fu teaches the limitations recited in claims 4 and 18. For example, Appellants are unable to determine what specific teaching in Fu is being relied upon to teach the claimed "inducing a placebo error condition in close proximity to code for allocating said resource."

The Examiner's responded to these arguments on page 13 of the Second Office Action, in which the Examiner asserted the following:

> Fu teaches teach inducing a placebo error condition in close proximity to code for allocating said resource (Fu, EIG. 5, step 600, FIG. 6, steps 610, 620, 630, [0042]-[0047], "in decision block 530, a determination is made whether minima point processing subroutine 600 returned an indication of a memory leak. If a memory leak was found, processing proceeds to block 599 where the memory usage processing subrouting 500 ends and a memory leak message is returned to the calling routing" lines 17-25, and [0045], "subroutine 600 proceeds to decision block 610 where a test is made to determine whether at least four memory usage data minima point were found, If less that tour memory leak message data minima points were found, processing proceeds to block 699, where subroutine 600 ends and a memory leak message is returned to the calling routine"); Fu teaches the steps 610, 620, and 630 in FIG. 6, read in the limitation of "placebo error condition" in claims 4 and 18 in the present application. (emphasis added)

Despite the Examiner's assertion in the above-underlined passage, Appellants are still unclear as to why "steps 610, 620, and 630" teach the claimed "inducing a placebo error condition in close proximity to code for allocating said resource." The teachings in Fu associated with the Examiner's cited passages are completed unrelated to this limitation. In this regard, Appellants note that the Examiner has failed to set forth any claim construction, either for the entire

limitation or for the phrase "placebo error condition."

In paragraph [0043], Fu teaches that "minima point processing subroutine 600 analyzes the minima points of the memory usage data of a particular memory usage record to determine if the minima points do or do not indicate a memory leak in conjunction with the previous determination that the area under first and second derivative curves exceeds zero." The subroutine 600 includes steps 610, 620, and 630. Again, how these teachings relate to the limitations recited in claim 4 is entirely unclear to Appellants. Therefore, Appellants maintain the argument that the Examiner has failed to establish that Fu teaches the limitations for which Fu is being relied upon to teach.

Conclusion

Based upon the foregoing, Appellants respectfully submit that the Examiner's rejections under 35 U.S.C. §§ 101-103 is not viable. Appellants, therefore, respectfully solicit the Honorable Board to reverse the Examiner's rejections under 35 U.S.C. §§ 101-103.

To the extent necessary, a petition for an extension of time under 37 C.F.R. § 1.136 is hereby made. Please charge any shortage in fees due under 37 C.F.R. §§ 1.17, 41.20, and in connection with the filing of this paper, including extension of time fees, to Deposit Account 09-0461, and please credit any excess fees to such deposit account.

Date: October 24, 2007

Respectfully submitted,

/Scott D. Paul/

Scott D. Paul
Registration No. 42,984
Steven M. Greenberg
Registration No. 44,725
Phone: (561) 922-3845
CUSTOMER NUMBER 46320

VIII. CLAIMS APPENDIX

1. A memory leak detection and reporting method comprising the steps of:

time stamping allocated ones of resources in a resource pool;

identifying calling code segments receiving said allocated resources;

detecting memory leaks by inspecting individual timestamps for said allocated resources to determine whether said allocated resources have become overly idle; and,

for each allocated resource determined to have become overly idle, reporting an identity of a corresponding one of said calling code segments.

2. The method of claim 1, wherein said identifying step comprises the step of retrieving identities for individual ones of said calling code segments from an associated calling stack when said individual ones of said calling code segments acquire one of said allocated resources.

3. The method of claim 1, further comprising the step of performing said detecting and reporting steps in a separate thread of execution.

4. The method of claim 2, wherein said retrieving step comprises the steps of:

for calling code segment in which a resource is allocated, inducing a placebo error condition in close proximity to code for allocating said resource; and,

reading error data from said calling stack produced by said placebo error condition, said error data comprising identity information for said calling code segment.

5. The method of claim 1, further comprising the step of performing said detecting and reporting steps responsive to allocating one of said resources in said resource pool.

6. The method of claim 1, further comprising the step of performing said detecting and reporting steps responsive to an elapsing interval.

7. A memory leak detection and reporting system comprising:

a resource pool comprising a plurality of allocable resources;

a pool manager programmed to manage allocation of said allocable resources to calling code segments; and,

a data store of allocated resources and corresponding identities for calling code segments receiving said allocated resources, wherein

the pool manager

detects memory leaks by inspecting individual timestamps for said allocated resources to determine whether said allocated resources have become overly idle; and,

for each allocated resource determined to have become overly idle, reports an identity of a corresponding one of said calling code segments to the data store.

8. The system of claim 7, wherein said pool manager comprises:

a communicative coupling to a call stack; and,

correlation logic for correlating a calling code segment reference disposed in said call stack to a concurrently allocated one of said allocable resources.

9. The system of claim 7, further comprising a garbage collector coupled to said resource pool.

10. A memory leak detection and reporting method comprising the steps of:

allocating a resource from a resource pool, time stamping said allocated resource and recording an identity for a calling code segment acquiring said allocated resource;

updating said time stamp when said allocated resource is accessed;

inspecting said time stamp to determine if said allocated resource has become overly idle; and,

if it is determined that said allocated resource has become overly idle, reporting a suspected memory leak in association with said allocated resource and further reporting said recorded identity for said calling code segment which had acquired said allocated resource.

11. The method of claim 10, further comprising the step of performing said inspecting and reporting steps in a separate thread of execution.

12. The method of claim 10, further comprising the steps of performing said allocating, inspecting and reporting steps in a pool manager.

13. The method of claim 10, further comprising the step of performing said inspecting and reporting steps responsive to allocating another resource in said resource pool.

14. The method of claim 10, further comprising the step of performing said inspecting and reporting steps responsive to an elapsing interval.

15. A machine readable storage having stored thereon a computer program for memory leak detection and reporting, the computer program comprising a routine set of instructions which when executed by the machine cause the machine to perform the steps of:

time stamping allocated ones of resources in a resource pool;

identifying calling code segments receiving said allocated resources;

detecting memory leaks by inspecting individual timestamps for said allocated resources to determine whether said allocated resources have become overly idle; and,

for each allocated resource determined to have become overly idle, reporting an identity of a corresponding one of said calling code segments.

16. The machine readable storage of claim 15, wherein said identifying step comprises the step of retrieving identities for individual ones of said calling code segments from an associated calling stack when said individual ones of said calling code segments acquire one of said allocated resources.

17. The machine readable storage of claim 15, further comprising the step of performing said detecting and reporting steps in a separate thread of execution.

18. The machine readable storage of claim 16, wherein said retrieving step comprises the steps of:

for calling code segment in which a resource is allocated, inducing a placebo error condition in close proximity to code for allocating said resource; and,

reading error data from said calling stack produced by said placebo error condition, said error data comprising identity information for said calling code segment.

19. The machine readable storage of claim 15, further comprising the step of performing said detecting and reporting steps responsive to an elapsing interval.

20. A machine readable storage having stored thereon a computer program for memory leak detection and reporting, the computer program comprising a routine set of instructions which when executed by the machine cause the machine to perform the steps of:

allocating a resource from a resource pool, time stamping said allocated resource and recording an identity for a calling code segment acquiring said allocated resource;

updating said time stamp when said allocated resource is accessed;

inspecting said time stamp to determine if said allocated resource has become overly idle; and,

if it is determined that said allocated resource has become overly idle, reporting a suspected memory leak in association with said allocated resource and further reporting said recorded identity for said calling code segment which had acquired said allocated resource.

IX. EVIDENCE APPENDIX

No evidence submitted pursuant to 37 C.F.R. §§ 1.130, 1.131, or 1.132 of this title or of any other evidence entered by the Examiner has been relied upon by Appellants in this Appeal, and thus no evidence is attached hereto.

X. RELATED PROCEEDINGS APPENDIX

Since Appellants are unaware of any related appeals and interferences, no decision rendered by a court or the Board is attached hereto.